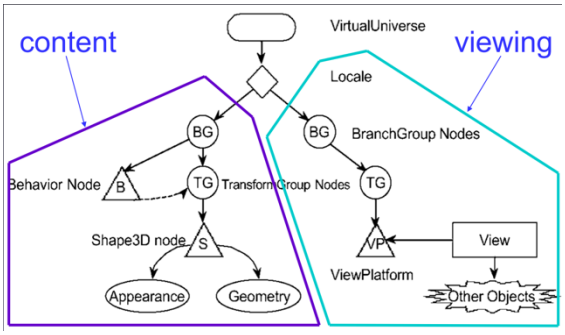
Grundlagen der Computergraphik

VO-Prüfungsvorbereitung (gestellte Fragen im 2023W)

Datenstrukturen

Was ist ein Szenen Graph? Welche Extraknoten werden in Unity dafür benötigt?



Ein Szenengraph ist eine Datenstruktur, die verwendet wird, um die logische und oft auch die räumliche Organisation einer grafischen Szene zu verwalten. Er stellt die Szene als eine Hierarchie von Knoten dar, wobei jeder Knoten ein Objekt oder eine Gruppe von Objekten repräsentiert, die in der Szene dargestellt werden sollen. Diese Knoten können verschiedene Dinge darstellen, wie geometrische Formen, Kameras, Lichtquellen, Gruppierungen anderer Objekte (für organisatorische Zwecke) und sogar Operationen wie Transformationen, die auf ein Objekt oder eine Gruppe von Objekten angewendet werden sollen.

In einem Szenengraphen repräsentiert die Wurzel des Graphen die gesamte Szene, während die Blätter die detailliertesten Einheiten in der Szene darstellen, wie einzelne geometrische Objekte. Die Struktur eines Szenengraphen ermöglicht es, komplexe Szenen effizient zu verwalten, zu rendern und zu manipulieren, indem Transformationen und andere Operationen rekursiv von den Elternknoten zu ihren Kindknoten angewendet werden.

Extraknoten in Unity

In Unity werden Szenen durch die Hierarchie von „GameObjects“ und deren „Components“ strukturiert. Für die Erstellung eines Szenengraphen in Unity könnten folgende spezielle Knoten (bzw. GameObjects und Components) erforderlich sein:

Transform-Komponente:
Die Transform-Komponente ist essenziell und vorhanden in jedem GameObject. Sie repräsentiert die Transformation (Position, Rotation, Skalierung) des Objekts im Raum. Transformationen sind hierarchisch, was bedeutet, dass die Transformation eines Elternobjekts auf all seine Kindobjekte angewendet wird.

Mesh Renderer und Mesh Filter (für Geometrie):

- *Mesh Filter:*
  Diese Komponente speichert die Geometrie (Mesh) eines Objekts. Ein Mesh ist im Wesentlichen eine Sammlung von Punkten im 3D-Raum (Vertices), die definieren, wie ein Objekt geformt ist.
- *Mesh Renderer:*
  Bestimmt, wie das Mesh, das im Mesh Filter definiert ist, auf dem Bildschirm dargestellt wird. Es ist verantwortlich für das Zeichnen des Meshes an der Position, die durch die Transform-Komponente des GameObjects festgelegt wird.

Materialien (für Material):
Materialien in Unity definieren das Aussehen von Oberflächen. Jedes Material verwendet einen Shader, der bestimmt, wie das Material unter verschiedenen Lichtbedingungen und aus verschiedenen Blickwinkeln aussieht. Materialien werden dem Mesh Renderer zugeordnet, um das endgültige Erscheinungsbild der Geometrie zu bestimmen.

Zusätzlich zu diesen Kernkomponenten können weitere spezialisierte Komponenten oder GameObjects je nach den spezifischen Anforderungen der Szene hinzugefügt werden, wie Lichtquellen, Kameras, und spezifische Skripte für Verhaltensweisen oder Interaktionen.

Was ist ein Vertex und Fragment Shader? Wie viele Zugriffe braucht man, wenn man einen Ausschnitt von 800x600px hat?

Vertex Shader

Ein Vertex Shader ist die erste Stufe der Shader-Verarbeitung in der Grafikpipeline. Er bearbeitet jeden Vertex (Eckpunkt) eines 3D-Modells individuell. Die Hauptaufgaben eines Vertex Shaders umfassen die Transformation der Vertices von lokalen Koordinaten (wie sie im Modell definiert sind) in Clip-Koordinaten (ein koordiniertes System, das für den weiteren Rendering-Prozess genutzt wird), die Berechnung der Beleuchtung auf Vertex-Ebene und die Vorbereitung von Daten für den nächsten Schritt in der Pipeline, den Fragment Shader.

Fragment Shader

Ein Fragment Shader bearbeitet Fragmente, die potenzielle Pixel auf dem Bildschirm darstellen. Für jedes Fragment berechnet der Shader die endgültige Farbe und andere Attribute (wie Texturkoordinaten und Beleuchtung), basierend auf den Daten, die vom Vertex Shader übergeben wurden, und den im Shader definierten Algorithmen. Der Fragment Shader wird für jedes Fragment aufgerufen, das durch das Rasterisierungsverfahren generiert wird, welches die durch den Vertex Shader transformierten Vertices in eine 2D-Darstellung der Geometrie auf dem Bildschirm umwandelt.

Anzahl der Zugriffe

Der Vertex Shader wird für jeden Vertex eines zu zeichnenden Objekts aufgerufen. Die Anzahl der Aufrufe hängt also von der Komplexität der Szene ab, d.h. von der Anzahl der Vertices, die in allen zu zeichnenden Objekten vorhanden sind. Diese Zahl ist nicht direkt mit der Auflösung des Outputs (in diesem Fall 800x600) verbunden, sondern mit der Anzahl und Komplexität der 3D-Modelle in der Szene.

Der Fragment Shader hingegen wird für jedes Pixel aufgerufen, das potenziell im finalen Bild gezeichnet wird. Bei einer Auflösung von 800x600 Pixeln:

- Jeder Pixel auf dem Bildschirm wird mindestens einmal vom Fragment Shader verarbeitet, vorausgesetzt, die gesamte Szene bedeckt den gesamten Bildschirm.
- Daher gibt es 800 * 600 = 480.000 Aufrufe des Fragment Shaders.

Diese Zahl kann jedoch steigen, abhängig von Faktoren wie:

- *Überlappung von Objekten:*
  Wenn Objekte sich überlappen, kann für dieselben Pixel mehr als einmal ein Fragment Shader Aufruf erfolgen, bevor der Z-Buffer (oder eine ähnliche Technik) verwendet wird, um zu bestimmen, welches Fragment sichtbar bleibt.
- *Post-Processing-Effekte:*
  Manche Effekte erfordern zusätzlichen Passes über den gesamten Bildschirm, was bedeutet, dass der Fragment Shader zusätzlich für jeden Pixel aufgerufen wird.

Was ist ein Octree? Was sind die Vor- und Nachteile eines Octrees und was ist der Unterschied zu einem Grid?

Ein Octree ist eine Baumstruktur, die verwendet wird, um den dreidimensionalen Raum effizient zu partitionieren. Jeder Knoten im Baum repräsentiert einen Würfel (oder einen "Block") im Raum, und jeder dieser Knoten hat bis zu acht Kinder. Jedes Kind repräsentiert einen der acht Unterwürfel, in die der Würfel des Elternknotens geteilt wird. Diese rekursive Aufteilung ermöglicht eine sehr effiziente Organisation und Abfrage von räumlichen Daten, insbesondere wenn die Daten ungleichmäßig im Raum verteilt sind.

Vorteile eines Octrees:

- *Effizienz bei der räumlichen Suche:*
  Octrees ermöglichen eine schnelle Suche und Zugriff auf räumliche Daten, da die Baumstruktur das Ausschließen großer, irrelevanter Raumbereiche ermöglicht, ohne jeden Punkt einzeln prüfen zu müssen.
- *Anpassungsfähigkeit an komplexe Strukturen:*
  Durch die rekursive Unterteilung kann der Octree detaillierte Bereiche des Raums feiner aufteilen, während weniger detaillierte Bereiche gröber bleiben. Dies führt zu einer effizienteren Speichernutzung im Vergleich zu einer gleichmäßigen Gitterstruktur.
- *Dynamische Anpassung:*
  Octrees können leicht modifiziert werden, um sich an Änderungen in der räumlichen Verteilung der Daten anzupassen, ohne die gesamte Struktur neu aufbauen zu müssen.

Nachteile eines Octrees:

- *Speicherüberhang:*
  Jeder Knoten im Baum benötigt zusätzlichen Speicher für Verwaltungsinformationen, was bei sehr großen Datensätzen zu einem nicht unerheblichen Speicherüberhang führen kann.
- *Ungleichmäßige Leistung:*
  Die Leistung von Operationen kann je nach Struktur des Baumes und der Verteilung der Daten im Raum variieren.
- *Komplexität in Implementierung und Verwaltung:*
  Die Verwaltung eines Octrees, insbesondere das Einfügen und Löschen von Knoten, kann komplexer sein als die Verwaltung einfacher Gitterstrukturen.

Unterschied zu einem Grid:

Ein Grid (oder ein gleichmäßiges Gitter) teilt den Raum in eine regelmäßige Anordnung von Zellen (oder "Voxeln") gleicher Größe. Dieser Ansatz ist sehr direkt und einfach zu implementieren, hat aber den Nachteil, dass er möglicherweise viel Speicher für Bereiche des Raums verwendet, die keine oder nur wenige Daten enthalten.

Speichernutzung:
Im Gegensatz zu einem Octree, der dynamisch nur die Bereiche des Raumes unterteilt, die Daten enthalten, reserviert ein Grid Speicher für jeden Voxel, unabhängig davon, ob er Daten enthält oder nicht.

Suchleistung:
Während Octrees eine schnelle Ausschlussmethode für leere Raumbereiche bieten, müssen bei einem Grid möglicherweise viele leere Zellen durchsucht werden, was zu einer geringeren Effizienz führen kann, besonders wenn die Daten ungleichmäßig verteilt sind.

Flexibilität:
Grids sind weniger flexibel bei der Anpassung an die räumliche Dichte der Daten, da alle Zellen dieselbe Größe haben.

Was ist ein BSP-Tree? (Transformationen, Kombinationen, Vor- und Nachteile, Unterschied zu kDTree)

Ein BSP-Baum (Binary Space Partitioning-Baum) wird verwendet, um Szenen zu partitionieren. Der Raum wird dabei rekursiv durch Ebenen in konvexe Teilräume (Partitions) aufgeteilt. Jeder Knoten im BSP-Baum repräsentiert eine Teilungsebene und teilt die Welt in zwei Halbräume: den "vorderen" und den "hinteren" Halbraum.

Transformationen

- BSP-Bäume sind effektiv bei der Handhabung von Transformationen, da sie das Hinzufügen und Entfernen von Objekten durch die Anpassung der Baumstruktur ermöglichen, ohne den gesamten Baum neu zu berechnen.
- Transformationen wie Rotation und Skalierung können aufgrund der binären Natur des Baumes und der damit verbundenen räumlichen Partitionierung komplexer sein, besonders wenn sie eine Neuberechnung der Teilungsebenen erfordern.

Kombinationen

- *Kombination mit B-Rep, dann neuer BSP-Baum*
  In diesem Ansatz wird zunächst eine B-Rep-Darstellung eines Objekts oder einer Szene erstellt. Dann wird ein BSP-Baum basierend auf dieser Darstellung generiert.
- *Direkte Kombination von BSP-Bäumen:*
  Bei dieser Methode werden zwei oder mehr bestehende BSP-Bäume direkt kombiniert, um einen neuen BSP-Baum zu erstellen. Dies ist besonders nützlich, wenn man bereits BSP-Bäume für verschiedene Teile der Szene hat und diese kombinieren möchte, ohne von Grund auf neu zu beginnen.

Vorteile

- *Effizientes Rendering:*
  BSP-Bäume sind besonders nützlich für das Rendering, da sie schnelle Sichtbarkeitstests ermöglichen, indem sie Szenen entsprechend der Position des Betrachters aufteilen.
- *Kollisionserkennung und Raytracing:*
  Sie sind effektiv bei Kollisionserkennung und Raytracing, da sie eine binäre Suche entlang der Raypaths ermöglichen.

Nachteile

- *Speicherintensität:*
  BSP-Bäume können speicherintensiv sein, da sie Informationen für jede Teilungsebene speichern müssen.
- *Rechenintensive Vorbereitung:*

Das Erstellen eines BSP-Baumes kann rechenintensiv sein, vor allem, wenn dynamische Szenen häufige Neuaufbauten erfordern.

- *Überlappende Polygone:*
  Die Handhabung von Polygonen, die Teilungsebenen überlappen, kann kompliziert sein und erfordert spezielle Behandlung.

Rendering

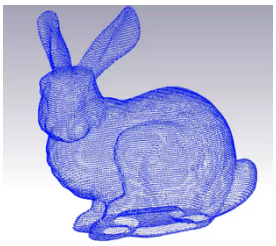
- BSP-Bäume können das *Painter's Algorithm-Rendering* ermöglichen, bei dem Szenen von hinten nach vorne gerendert werden, um korrekte Überlappungen ohne Tiefentests zu gewährleisten.
- Sie erlauben "Back-Face Culling", indem nur diejenigen Geometrien gerendert werden, die in Richtung der Kamera orientiert sind.

Vergleich mit kd-Bäumen

Sowohl BSP-Bäume als auch kd-Bäume sind raumteilende Datenstrukturen, die verwendet werden, um räumliche Szenen zu organisieren und Suchvorgänge zu optimieren. kd-Bäume sind eine spezialisierte Form von BSP-Bäumen, die für Punktwolken und andere raumbezogene Daten optimiert sind. Sie teilen den Raum entlang der *Koordinatenachsen* statt beliebiger Ebenen.

BSP-Bäume werden oft in Rendering-Systemen für komplexe Szenen eingesetzt, während kd-Bäume typischerweise in Anwendungen wie Raytracing und Kollisionserkennung für ihre schnellen Suchfunktionen genutzt werden. kd-Bäume sind in der Regel besser für dynamische Szenen geeignet, wo Objekte sich bewegen, da sie einfacher anzupassen sind als BSP-Bäume.

Was ist eine Point Cloud?



Eine Point Cloud ist eine *Sammlung von Datenpunkten* im Raum. Jeder Punkt in der Cloud repräsentiert eine Position im dreidimensionalen Raum und wird oft durch seine X-, Y- und Z-Koordinaten dargestellt.

Erzeugung

Point Clouds können auf verschiedene Weise erzeugt werden. Eine gängige Methode ist die Verwendung von 3D-Scannern, die Laser oder strukturiertes Licht verwenden, um die Oberfläche eines physischen Objekts oder einer Szene zu erfassen. Die resultierenden Punkte bilden eine direkte Messung der Form und manchmal auch der Farbe und anderer Eigenschaften der Oberfläche.

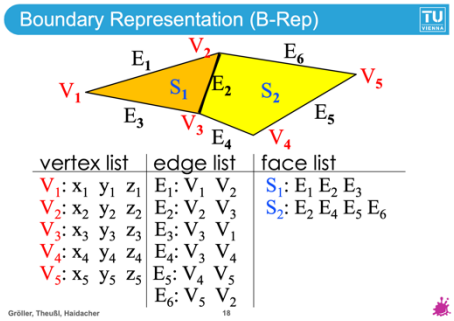
Herausforderungen

Die Verarbeitung von Point Clouds bringt einige Herausforderungen mit sich:

- *Datenmenge:*
  Point Clouds können aus Millionen von Punkten bestehen, welche die Verarbeitung, Speicherung und Analyse erschweren.

- *Rauschen und Unvollständigkeit:*
  Die Daten können Rauschen enthalten und oft nur die Oberfläche des Objekts von bestimmten Blickwinkeln erfassen, was zu Unvollständigkeiten führt.
- *Verarbeitung und Analyse:*
  Die Umwandlung von Point Clouds in nützliche Modelle oder die Extraktion spezifischer Informationen erfordert fortschrittliche Algorithmen und Software.

Was ist ein B-Rep? (Vor- und Nachteile, Kombinationen)



Ein B-Rep, oder Boundary Representation, ist eine Methode zur Darstellung von Formen oder Objekten in der computergestützten Geometrie und CAD (Computer-Aided Design). Im Gegensatz zu Volumen- oder Punktwolkenrepräsentationen, die ein Objekt durch sein Inneres oder durch eine Menge von Punkten im Raum definieren, beschreibt ein B-Rep die *Oberfläche eines Objekts*. Die grundlegende Idee besteht darin, die Grenzen eines Körpers – also Flächen, Kanten und Eckpunkte (Vertices) – explizit zu speichern, um die Form des Objekts zu definieren.

Ein B-Rep-Modell besteht typischerweise aus:

- *Vertices (Eckpunkte):*
  Die Punkte im 3D-Raum, an denen sich zwei oder mehr Kanten treffen.
- *Edges (Kanten):*
  Linienstücke, die zwei Vertices verbinden und den Übergang zwischen zwei Flächen darstellen.
- *Faces (Flächen):*
  Die zweidimensionalen Oberflächen, die von den Kanten begrenzt werden und die äußere Hülle des Objekts bilden.

Diese Elemente sind durch eine Reihe von Beziehungen miteinander verbunden, die definieren, wie Kanten Vertices verbinden und wie Kanten und Vertices zusammenkommen, um Flächen zu bilden. In einer typischen B-Rep-Datenstruktur werden diese Beziehungen durch *Indizes* dargestellt, die auf Listen oder Arrays von Vertices, Kanten und Flächen verweisen, anstatt direkte Zeiger auf die Objekte selbst zu verwenden.

Vorteile von B-Reps:

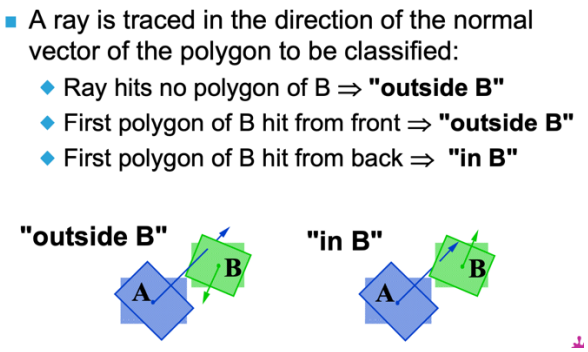
- *Allgemeine Darstellung:*
  B-Reps bieten eine sehr allgemeine Darstellungsform, was bedeutet, dass sie für eine Vielzahl von Objekten verwendet werden können, von einfachen bis hin zu sehr komplexen Formen.
- *Erzeugung von Modellen durch Digitalisierung:*
  B-Reps sind besonders nützlich, um digitale Modelle aus physischen Objekten zu erstellen. Durch Verfahren wie 3D-Scanning können real existierende Objekte präzise in ein digitales Format überführt werden.
- *Transformationen sind einfach und schnell:*
  Geometrische Transformationen wie Translation, Rotation und Skalierung können auf B-Reps effizient angewendet werden, da die Transformationen auf die Vertices angewandt und die Verbindungen (Edges und Faces) entsprechend aktualisiert werden können.

Nachteile von B-Reps:

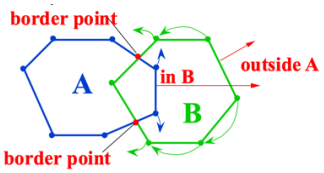
- *Hoher Speicherbedarf:*
  B-Reps können speicherintensiv sein, da für jede Fläche, Kante und jeden Vertex Informationen gespeichert werden müssen.
- *Kombinationen sind relativ aufwändig:*
  Wenn B-Reps mit anderen geometrischen Darstellungen kombiniert werden, zum Beispiel mit Constructive Solid Geometry (CSG) für komplexe Operationen, kann dies rechenintensiv sein.
- *Gekrümmte Objekte müssen angenähert werden:*
  B-Reps müssen gekrümmte Oberflächen mit einer Menge von Flächen annähern, was die Genauigkeit beeinträchtigen und zu einer größeren Datenmenge führen kann.

Kombinationen und Verbesserungen in der Praxis:

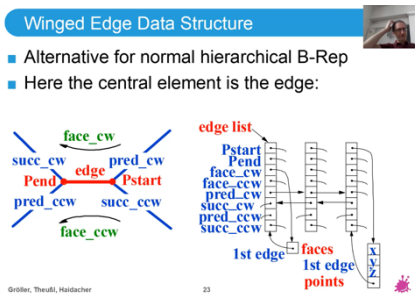
- Box Enclosures for Simple Testing:
  Indem man jedem Polygon eine umschließende Box zuweist, kann man schnell und einfach feststellen, ob sich zwei Polygone überschneiden könnten, was die Berechnung von Kollisionen oder Interferenzen vereinfacht.
- *Nur konvexe Polygone verwenden:*
  Wenn man sich auf konvexe Polygone beschränkt, werden die Tests auf Überschneidungen einfacher, da konvexe Polygone einfacher zu handhaben sind als konkave.
- *Strahlenklassifizierung zur Bestimmung der Position:*
  Um zu bestimmen, ob ein Punkt innerhalb oder außerhalb eines Körpers liegt, kann ein Strahl in Richtung der Normalenvektoren verfolgt werden. Wie der Strahl die Polygone trifft (von vorne oder von hinten), bestimmt die Klassifizierung des Punktes.



- ***Verbesserung*** *- Markierung von Grenzpunkten während der Teilung:*
  Eine Verbesserung der Effizienz kann erreicht werden, indem man während des Teilungsprozesses Grenzpunkte kennzeichnet. Dies reduziert die Anzahl der Polygone, die durch komplexere Methoden klassifiziert werden müssen.



Was ist eine Winged Edge Data Stucture?



Das zentrale Element der Winged Edge Datenstruktur ist die Kante (Edge). Für jede Kante werden Informationen gespeichert, die es ermöglichen, schnell auf angrenzende Kanten, Flächen und Vertices zuzugreifen. Die "Winged" Bezeichnung bezieht sich auf die Möglichkeit, von einer Kante aus in vier "Richtungen" zu navigieren:

- Zu den beiden Endvertices der Kante.
- Zu den beiden angrenzenden Flächen (Faces), die von der Kante getrennt werden.

Für jede Kante in einem Mesh speichert die Winged Edge Datenstruktur typischerweise Folgendes:

- *Vertex-Referenzen:* Die Indizes oder Referenzen der beiden Endvertices.
- *Face-Referenzen:* Die Indizes oder Referenzen der beiden angrenzenden Flächen.
- *Edge-Referenzen:* Für jeden Endvertex speichert die Struktur die Referenzen auf die vorherige und die nächste Kante in der Umkreisung jeder der beiden angrenzenden Flächen.

Vorteile

- *Effiziente Navigation:*
  Ermöglicht schnellen Zugriff auf die zu einer Kante angrenzenden topologischen Elemente, was für viele geometrische und topologische Operationen nützlich ist.
- *Unterstützt komplexe Operationen:*
  Ermöglicht das effiziente Durchführen von Operationen wie Edge-Split, Edge-Collapse, und Face-Traversal, die in der Modellierung und Bearbeitung von 3D-Meshes häufig benötigt werden.

Nachteile

- *Speicherbedarf:*
  Speichert mehr Informationen pro Kante als einfachere Datenstrukturen, was zu einem höheren Speicherverbrauch führen kann.
- *Komplexität:*
  Die Verwaltung und Aktualisierung der Datenstruktur kann komplex sein, insbesondere bei Modifikationen des Meshes, die die Topologie ändern (z.B. beim Hinzufügen oder Entfernen von Vertices, Kanten oder Flächen).

Modelling

Was sind Partikelsysteme?
Wie funktionieren Sie und wozu werden sie verwendet?

Partikelsysteme werden verwendet, um Effekte zu erzeugen, die aus vielen kleinen Teilchen bestehen, wie Rauch, Feuer, Funken, Wolken, Staub, Wasserfälle oder magische Effekte. Diese Systeme bieten eine flexible Methode, um komplexe und oft chaotische Phänomene zu simulieren, die schwer mit traditionellen Rendering-Techniken darzustellen sind.

Wie Partikelsysteme funktionieren:

1. Initiierung:
Ein Partikelsystem wird mit einer Menge von Partikeln initialisiert. Jedes Partikel hat Attribute wie Position, Geschwindigkeit, Farbe, Lebensdauer, Größe und manchmal auch Rotation.

2. Simulation:
Die Bewegung und Veränderung der Partikel über die Zeit werden simuliert, oft unter Einbeziehung von physikalischen Kräften wie Schwerkraft, Wind oder Turbulenzen. Die Eigenschaften der Partikel können sich im Laufe ihrer Lebensdauer ändern, zum Beispiel können sie verblassen, ihre Farbe ändern oder schrumpfen.

3. Rendering:
Die Partikel werden auf dem Bildschirm gerendert. Dies kann durch einfache Punkte, Sprites (2D-Texturen), Volumen oder andere geometrische Formen geschehen. Die Darstellung kann von der Entfernung zum Betrachter, der Perspektive und anderen visuellen Effekten beeinflusst werden.

4. Recycling:
Wenn Partikel ihre Lebensdauer beendet haben oder eine bestimmte Bedingung erfüllen (z.B. den Boden berühren), können sie entfernt oder recycelt werden, indem ihre Eigenschaften zurückgesetzt und sie erneut in das System eingeführt werden.

Durch ihre Vielseitigkeit und die Fähigkeit, komplexe visuelle Phänomene zu simulieren, sind Partikelsysteme ein unverzichtbares Werkzeug in der Computergrafik und darüber hinaus geworden.

Erkläre den Unterschied zwischen implizite, explizite, parametrische und prozedurale Modellierung

Explizite Modellierung

In der expliziten Modellierung wird jedem Wert von x genau ein Wert von y zugeordnet, oft in Form einer Funktion $y = f(x)$. Dies ist eine direkte Art der Modellierung, bei der die Position jedes Punktes direkt durch die Funktion bestimmt wird. Explizite Modelle sind einfach zu verstehen und zu implementieren, haben aber den Nachteil, dass sie nicht immer komplexere Formen oder Topologien leicht darstellen können.

Beispiel: Eine einfache Kurve in 2D, wie $y = x^2$, ist ein Beispiel für explizite Modellierung.

Implizite Modellierung

Implizite Modellierung beschreibt Objekte durch eine Relation zwischen Variablen, typischerweise in Form einer Gleichung $f(x, y, z) = 0$. Anstatt Punkte direkt zu spezifizieren, definiert sie eine Oberfläche als die Menge aller Punkte (x, y, z), die die Gleichung erfüllen. Dies ermöglicht die Darstellung komplexer Oberflächen und Formen, die schwer explizit zu modellieren sind, wie zum Beispiel Objekte mit Hohlräumen oder solche, die sich selbst schneiden.

Beispiel: Eine Kugel kann implizit als $x^2 + y^2 + z^2 - r^2 = 0$ modelliert werden, wobei r der Radius der Kugel ist.

Parametrische Modellierung

Parametrische Modellierung definiert Geometrien durch Parametergleichungen. Anstatt eine direkte Zuordnung von x zu y vorzunehmen (wie in der expliziten Modellierung) oder eine Bedingung zu setzen (wie in der impliziten Modellierung), steuern Parameter die Positionen der Punkte im Raum. Dies erlaubt eine flexible Kontrolle über die Form der Objekte und macht es einfach, Veränderungen durch Anpassung der Parameter vorzunehmen.

Beispiel: Ein Kreis in der parametrischen Form kann durch die Gleichungen $x = r * cos(t)$ und $y = r * sin(t)$ beschrieben werden, wobei t ein Parameter ist, der den Winkel angibt, und r der Radius des Kreises ist.

Prozedurale Modellierung

Prozedurale Modellierung ist eine Methode in der Computergrafik, bei der Algorithmen und Regeln genutzt werden, um komplexe Modelle und Umgebungen automatisch zu erzeugen. Anstatt jedes Detail manuell zu erstellen, definiert man, wie Objekte basierend auf Parametern aussehen sollen. Das ermöglicht die schnelle Erstellung von vielfältigen und detaillierten Szenen wie Landschaften, Städten oder Texturen mit relativ wenig Aufwand.

TLDR:

- *Explizite Modellierung:*
  Direkte Zuordnung von x zu y; einfach, aber eingeschränkt in der Komplexität.
- *Implizite Modellierung:*
  Definiert durch eine Bedingung, die Punkte im Raum erfüllen müssen; gut für komplexe Oberflächen, schwerer zu berechnen.
- *Parametrische Modellierung:*
  Nutzt Parameter, um Punktpositionen zu steuern; bietet hohe Flexibilität und Kontrolle, ideal für die Darstellung komplexer Formen und Kurven.
- *Prozedurale Modellierung:*
  Während implizite und parametrische Modellierung sich auf die mathematische Beschreibung der Formen konzentrieren, und explizite Modellierung direkte Zuordnungen nutzt, fokussiert

sich prozedurale Modellierung auf die Erstellung von Regeln und Algorithmen, die die Komplexität und Details der Modelle steuern.

Was sind Sweeps? Erkläre Vor- und Nachteile.

"Sweep" (oder Sweeping) bezeichnet eine Modellierungstechnik, bei der ein 2D-Profil oder eine Form entlang eines Pfades oder einer Achse bewegt (gefegt) wird, um ein 3D-Objekt zu erstellen. Diese Methode wird oft verwendet, um Objekte mit komplexen oder symmetrischen Formen zu modellieren, die sich entlang einer bestimmten Trajektorie entwickeln.

Vorteile von Sweeps:

- *Komplexe Formen:*
  Ermöglicht die Erstellung von komplexen und detaillierten 3D-Objekten, die mit herkömmlichen Modellierungstechniken schwierig zu realisieren wären.
- *Symmetrische Objekte:*
  Ideal für die Modellierung von symmetrischen oder gleichmäßig geformten Objekten, da das 2D-Profil entlang eines vordefinierten Pfades gefegt wird.
- *Effizienz:*
  Kann effizienter sein, um bestimmte Arten von Objekten zu modellieren, da nur das Profil und der Pfad definiert werden müssen, nicht das gesamte Objekt.

Nachteile von Sweeps:

- *Modellierungskomplexität:*
  Das Erstellen des initialen Profils und des Pfades kann kompliziert sein, besonders wenn die gewünschte Trajektorie des Sweeps komplex ist.
- *Rendering-Schwierigkeiten:*
  Objekte, die durch Sweeping erstellt wurden, können aufgrund ihrer komplexen Geometrien und Oberflächen schwer zu rendern sein, was zu höheren Rechenzeiten oder speziellen Rendering-Herausforderungen führen kann.
- *Eingeschränkte Flexibilität:*
  Obwohl Sweeps für bestimmte Formen ideal sind, können sie bei sehr unregelmäßigen oder organischen Formen, die nicht gut durch eine einfache Bewegung eines Profils entlang eines Pfades beschrieben werden können, Einschränkungen aufweisen.

Sweeps sind eine mächtige Technik, die für spezifische Anwendungsfälle wie die Erstellung von Rohren, Vasen, Möbeln und anderen Objekten mit klaren, geführten Formen besonders geeignet ist.

Was ist Cellular Texture Generation?

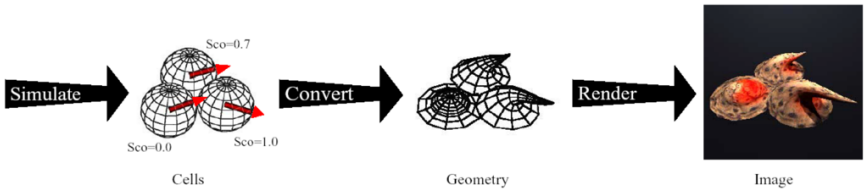
Cellular Texture Generation zielt darauf ab, Texturen und Oberflächenstrukturen durch die Simulation von zellulären Prozessen zu erzeugen. Dieser Ansatz basiert auf der Idee, dass die visuellen Merkmale vieler natürlicher Materialien und Oberflächen – wie Fell, Gras, Haut oder sogar organische Gewebe – durch die Interaktionen und das Verhalten von Zellen und ihrer Umgebung auf mikroskopischer Ebene bestimmt werden.

Die grundlegenden Komponenten dieses Ansatzes umfassen:

Zellzustände:
Diese beinhalten Eigenschaften wie Position, Orientierung, Form und chemische Konzentrationen innerhalb jeder Zelle. Diese Zustände beeinflussen, wie sich die Zellen verhalten und wie sie mit ihrer Umgebung interagieren.

Zellprogramme:
Das sind regelbasierte Anweisungen, die das Verhalten der Zellen steuern. Beispiele für solche Verhaltensweisen sind Bewegung zur Oberfläche, Zelltod bei zu großer Entfernung von der Oberfläche, Ausrichtung an anderen Zellen, Anhaften an benachbarte Zellen und Zellteilung, bis eine Oberfläche vollständig bedeckt ist.

Extrazelluläre Umgebungen:
Dazu gehören Faktoren wie die Orientierung der Nachbarzellen, die die Interaktionen zwischen Zellen und ihre Anordnung beeinflussen.



Ein spezifisches Beispiel ist die Simulation von Fell. Hierbei können Zellprogramme genutzt werden, um das Wachstum, die Ausrichtung und die Dichte der Fellhaare zu simulieren, basierend auf den zugrunde liegenden zellulären Prozessen und Umgebungsinteraktionen. Dies ermöglicht die Erzeugung realistischer Felltexturen für Tiere in Computerspielen, Filmen oder anderen visuellen Medien.

Erkläre wie Terrain Simulation funktioniert.

Terrain Simulation ist ein Prozess, bei dem digitale Landschaften und Geländeoberflächen für Anwendungen wie Spiele, Simulationen und visuelle Effekte erzeugt werden. Diese Simulationen nutzen verschiedene Techniken und Datenquellen, um realistische oder stilisierte Terrains zu modellieren. Hier sind die Schlüsselelemente und Methoden, die in der Terrain Simulation verwendet werden:

Fraktale
Fraktale sind eine Art von mathematischem Muster, das dazu verwendet wird, natürliche Landschaften wie Berge, Täler und Küstenlinien auf dem Computer zu erzeugen. Diese Muster sind besonders, weil sie sich selbst ähneln: Ein kleiner Teil eines Fraktals sieht ähnlich aus wie das ganze Fraktal, egal wie stark man heranzoomt. In der Terrain Simulation helfen Fraktale dabei, realistisch aussehende Landschaften zu erstellen, weil auch die Natur oft solche wiederholenden Muster zeigt. Wenn man also ein Gelände mit Fraktalen modelliert, kann man mit relativ einfachen mathematischen Regeln komplexe und detaillierte Landschaften erschaffen, die den echten Landschaften sehr ähnlichsehen.

Geographische Daten
Geographische Daten (z.B. aus Satellitenbildern oder topographischen Karten) werden verwendet, um reale Weltlandschaften in der Computergrafik nachzubilden. Diese Daten enthalten Informationen über Höhenunterschiede und können direkt in die Terrain Simulation einfließen, um genaue Repräsentationen spezifischer Orte zu erstellen.

Simulationen

- *Erosion:*
  Erosionssimulationen modellieren die Effekte von Wind, Wasser und anderen natürlichen Kräften auf die Landschaft über die Zeit. Dies trägt dazu bei, das Terrain realistischer zu gestalten, indem Merkmale wie Flusstäler, Schluchten und Sedimentablagerungen erzeugt werden.
- *Strahlverfolgung (Ray Casting):*
  Um die Sichtbarkeit und Beleuchtung der Landschaft zu berechnen, kann ein Strahl über die Oberfläche "geschossen" werden, um zu bestimmen, wo er die Oberfläche schneidet. Dies ist nützlich für die Erzeugung von Schatten und für Sichtbarkeitsanalysen.

Self-Thinning
Self-Thinning ist ein Konzept aus der Ökologie, das in Terrain Simulationen verwendet wird, um die Verteilung von Vegetation realistisch zu gestalten. Es besagt, dass innerhalb eines bestimmten Radius um einige Pflanzenarten herum andere Pflanzen nicht wachsen können, was zur Modellierung von realistischen Pflanzenverteilungen und zur Verhinderung der Überfüllung von Vegetation in simulierten Landschaften beiträgt.

Erkläre rationale Kurven genauer: Bézier Kurve, B-Splines, NURBS. Kann man mit B-Splines einen Kreis bilden?

Rationale Kurven sind ein wichtiges Werkzeug, um glatte und präzise Kurven und Oberflächen zu erzeugen. Zu den bekanntesten Beispielen gehören Bézier-Kurven, B-Splines und NURBS (Non-Uniform Rational B-Splines). Jeder dieser Kurventypen hat spezifische Eigenschaften und Anwendungen:

Bézier-Kurven
Bézier-Kurven werden durch einen Satz von Kontrollpunkten definiert. Die Kurve beginnt im ersten Kontrollpunkt und endet im letzten, wobei die dazwischen liegenden Kontrollpunkte die Form der Kurve beeinflussen, aber nicht notwendigerweise von der Kurve berührt werden.

B-Splines (Basis-Splines)
B-Splines erweitern das Konzept der Bézier-Kurven, indem sie eine Reihe von Kontrollpunkten verwenden, um eine Kurve zu formen, die glatter und flexibler ist als eine einzelne Bézier-Kurve. Im Gegensatz zu Bézier-Kurven, die als ein Segment definiert sind, können B-Splines aus mehreren Segmenten bestehen, wobei die Übergänge zwischen den Segmenten nahtlos sind. Dies macht sie nützlich für die Modellierung komplexerer Formen und Oberflächen.

NURBS (Non-Uniform Rational B-Splines)
NURBS sind eine noch erweiterte Form der B-Splines, die durch die Einführung von Gewichten und nicht-uniformen Knotenvektoren eine größere Flexibilität und Kontrolle bieten. Die "Rationalität" erlaubt es NURBS, sowohl Standard geometrische Formen (wie Geraden und Ebenen) als auch komplexe Freiformoberflächen exakt zu modellieren.

Können B-Splines einen Kreis bilden?
Reine B-Splines können einen Kreis nicht exakt darstellen, da sie auf polynomiellen Funktionen basieren und ein Kreis keine polynomielle Gleichung ist. Um einen Kreis oder andere nicht-polynomielle Formen genau zu modellieren, werden oft NURBS verwendet. NURBS erlauben durch ihre rationalen Komponenten und Gewichtung der Kontrollpunkte die exakte Darstellung von Kreisen und anderen komplexen geometrischen Formen.

Der Grund, warum polynomielle Gleichungen keinen Kreis exakt darstellen können, liegt in der Natur der Kreisgleichung selbst. Ein Kreis mit dem Mittelpunkt im Koordinatenursprung und einem Radius r wird durch die Gleichung $x^2 + y^2 = r^2$ beschrieben. Diese Gleichung ist nicht polynomiell in Bezug auf x und y zusammen, weil sie eine Gleichung zweiten Grades ist, in der x und y symmetrisch sind und quadriert werden, und es gibt keine Möglichkeit, sie als eine einfache Summe von Potenzen von x oder y auszudrücken.

Erkläre den Casteljau-Algorithmus und das Bernsteinpolynom genauer

Der „de Casteljau-Algorithmus“ wird verwendet, um Bézier-Kurven zu berechnen und darzustellen. Benannt nach Paul de Casteljau, der ihn bei Citroën entwickelte, nutzt der Algorithmus eine rekursive Teilung, um eine Bézier-Kurve in beliebiger Auflösung zu approximieren.

Funktionsweise des de Casteljau-Algorithmus:

1. *Kontrollpunkte:*
   Eine Bézier-Kurve wird durch eine Reihe von Kontrollpunkten definiert.
2. *Linear Interpolation:*
   Der Algorithmus führt lineare Interpolationen zwischen diesen Kontrollpunkten durch, beginnend mit den ursprünglichen Kontrollpunkten und dann sukzessive zwischen den interpolierten Punkten der vorherigen Schritte.
3. *Teilung:*
   In jedem Schritt werden die Kontrollpunkte in Teile aufgeteilt, was neue Punkte erzeugt, die näher an der tatsächlichen Kurve liegen.
4. *Rekursion:*
   Dieser Prozess wird rekursiv fortgesetzt, wobei die Anzahl der Teilschritte (und somit die Genauigkeit) typischerweise durch die gewünschte Genauigkeit der Kurvenapproximation bestimmt wird.
5. *Endresultat:*
   Nach genügend Iterationen konvergieren die interpolierten Punkte zur Bézier-Kurve, und der Punkt auf der Kurve für einen gegebenen Parameterwert t kann berechnet werden.

Bernsteinpolynome:
Bernsteinpolynome sind die mathematische Grundlage für Bézier-Kurven. Eine Bézier-Kurve vom Grad n wird durch eine Linearkombination von Bernsteinpolynomen vom Grad n dargestellt. Aus dem Casteljau-Algorithmus lassen sich die Bernstein-Polynome ableiten. Die Bernstein-Polynome sind:

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i} * t^i$$

Die Basisfunktion $B_i^n(t)$ gibt das Gewicht des i-ten Kontrollpunktes an der Stelle t auf der Bezier-Kurve an. Der binomische Koeffizient $\binom{n}{i}$ ist ein Faktor, der angibt, wie viele Möglichkeiten es gibt, i Elemente aus einer Menge von n Elementen ohne Wiederholung und ohne Berücksichtigung der Reihenfolge zu wählen. Der Ausdruck $(1-t)^{n-i}$ gibt das Gewicht der Punkte an, die zum Anfang der Kurve neigen, während t^i das Gewicht der Punkte zum Ende der Kurve hin angibt.

- n:
  Die Ordnung oder der Grad der Bézier-Kurve, und gleichzeitig die höchste Potenz in der Basisfunktion. Er bestimmt, wie viele Kontrollpunkte für die Bézier-Kurve verwendet werden, was wiederum beeinflusst, wie komplex die Kurve sein kann.
  Für eine Bézier-Kurve des Grades n gibt es $n+1$ Kontrollpunkte.
- i:
  Der aktuelle Index der Basisfunktion und des Kontrollpunktes. Für jede Bézier-Kurve gibt es mehrere Basisfunktionen, eine für jeden Kontrollpunkt. Der Index i läuft von 0 bis n, wobei jede Basisfunktion einen Einfluss des entsprechenden Kontrollpunktes auf die Form der Kurve darstellt.
- t:
  Der Parameter, der typischerweise zwischen 0 und 1 variiert. Man kann sich t als eine Art Zeitvorstellung vorstellen, die steuert, an welchem Punkt der Kurve man sich befindet, wenn man von 0 (Start der Kurve) zu 1 (Ende der Kurve) geht.

Sampling & Reconstruction

Was ist Sampling, Was ist Reconstruction?

Sampling bedeutet, kontinuierliche Signale zu diskreten Zeitpunkten zu messen oder zu erfassen. Stell dir vor, du hast eine glatte, fließende Kurve (ein kontinuierliches Signal), und du wählst in regelmäßigen Abständen Punkte auf dieser Kurve aus. Diese ausgewählten Punkte sind deine *Samples*. Das Ziel des Samplings ist es, genügend Informationen über das kontinuierliche Signal zu sammeln, sodass du es später aus diesen diskreten Punkten rekonstruieren kannst.

Reconstruction ist der Prozess, bei dem aus den diskreten *Samples* wieder ein kontinuierliches Signal erstellt wird. Da du zwischen den Samples keine Informationen hast, musst du eine Form der Interpolation verwenden, um zu schätzen, wie die Kurve zwischen den Punkten verläuft. Dies kann durch verschiedene Methoden erfolgen, wobei die Wahl des Rekonstruktionsfilters entscheidend ist.

Rekonstruktionsfilter

- ***Box-Filter:***
  Dieser Filter nimmt den Mittelwert der benachbarten Samples als Approximation für die Zwischenwerte. Im Frequenzraum entspricht dies einer *Sinc*-Funktion. Der Box-Filter kann zu einem "blockigen" Aussehen führen, da alle Samples gleich gewichtet werden, was zu einer abrupten Änderung zwischen den Samples führen kann.
- ***Tent-Filter (Dreiecksfilter):***
  Dieser Filter führt eine lineare Interpolation zwischen den Samples durch, was zu einer glatteren Kurve als beim Box-Filter führt. Der Tent-Filter gewichtet die näher am Interpolationspunkt liegenden Samples stärker als die weiter entfernten. Im Frequenzraum entspricht dies der Faltung mit der Sinc-Funktion zum Quadrat $sinc^2$.

Die Reconstruction zielt darauf ab, das ursprüngliche kontinuierliche Signal so genau wie möglich aus den diskreten Samples zu approximieren. Es ist ein Balanceakt zwischen der Erhaltung der originalen Signalinformation und der Minimierung der durch das Sampling eingeführten Artefakte.

Erkläre das Faltungstheorem.

Das Faltungstheorem verbindet zwei Operationen: die Faltung im Zeit- oder Ortsraum und die Multiplikation im Frequenzraum. Die Bedeutung des Theorems liegt in der Vereinfachung komplexer Berechnungen.

1. Faltung im Zeit- oder Ortsraum:

Die Faltung $f_1 * f_2$ zweier Funktionen f_1 und f_2 ist eine Operation, die eine neue Funktion erzeugt. Diese neue Funktion gibt an jedem Punkt die Summe der Produkte der überlappenden Werte der beiden Funktionen an, wobei eine der Funktionen umgekehrt und entlang der Achse verschoben wird.

2. Fourier-Transformation:

Die Fourier-Transformation F wandelt eine Funktion aus dem *Zeit- oder Ortsraum* in den *Frequenzraum* um. Die transformierte Funktion $F(f)$ beschreibt, wie viel von jeder Frequenzkomponente in der ursprünglichen Funktion vorhanden ist.

3. Multiplikation im Frequenzraum:

Die Gleichung $F(f_1) * F(f_2)$ besagt, dass die Fourier-Transformationen der beiden ursprünglichen Funktionen multipliziert werden. Im Frequenzraum ist die Multiplikation viel einfacher als die Faltung im Zeit- oder Ortsraum. Sie ergibt das *Frequenzspektrum* der gefalteten Funktion.

4. Das Faltungstheorem:

Die Gleichung $F(f_1 * f_2) = F(f_1) * F(f_2)$ drückt das *Faltungstheorem* aus: Die Fourier-Transformation der Faltung zweier Funktionen ist gleich der Multiplikation ihrer Fourier-Transformationen. Dies bedeutet, dass wir die komplizierte Faltungsoperation im Zeit- oder Ortsraum vermeiden und stattdessen eine einfachere Multiplikation im Frequenzraum durchführen können.

5. Inverse Fourier-Transformation:

Nachdem die Fourier-Transformation verwendet wurde, um zwei Funktionen im Frequenzraum zu multiplizieren, muss man die *inverse Fourier-Transformation* anwenden, um das Ergebnis zurück in den Zeit- oder Ortsbereich zu bringen: $F^{-1}(f_1 * f_2) = F^{-1}(f_1) * F^{-1}(f_2)$.

Zusammenfassend ermöglicht das Faltungstheorem, dass man, anstatt direkt im Zeit- oder Ortsraum zu falten, die Funktionen in den Frequenzraum überführt, dort die Multiplikation vornimmt und das Ergebnis dann zurücktransformiert. Dies ist in der Praxis oft viel effizienter.

Anwendung in der Computergraphik:

Ein einfacher und häufiger Fall von Faltung in der Computergrafik ist die Anwendung eines Weichzeichnungsfilters (Blur) auf ein Bild. Ein Weichzeichnungsfilter kann als eine Funktion betrachtet werden, die auf ein Bild angewendet wird, um es weniger scharf zu machen.

Hier die Schritte im Detail:

1. Definition der Funktionen:
Das Originalbild wird als eine Funktion f_1 betrachtet, und der Weichzeichnungsfilter wird als eine andere Funktion f_2 dargestellt.

2. Faltung im Ortsraum:
Wenn du den Filter direkt auf das Bild anwendest, würdest du eine Faltung im Ortsraum durchführen. Für jedes Pixel im Bild würdest du die umliegenden Pixel betrachten, sie mit den entsprechenden Werten des Filters multiplizieren und die Ergebnisse aufsummieren, um das neue, weichgezeichnete Pixel zu erhalten.

3. Vereinfachung durch das Faltungstheorem:
Statt die Faltung direkt im Bild durchzuführen, kannst du sowohl das Bild als auch den Filter in den Frequenzraum transformieren, indem du die Fourier-Transformation auf beide anwendest.

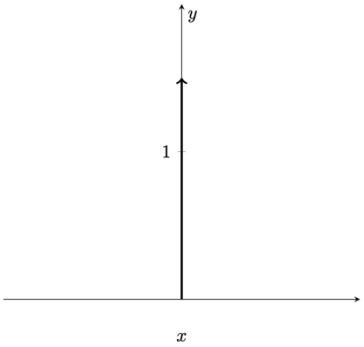
4. Multiplikation im Frequenzraum:
Im Frequenzraum multiplizierst du einfach die transformierten Funktionen miteinander. Dies ist vergleichsweise eine viel einfachere Operation.

5. Rückkehr zum Ortsraum:
Nach der Multiplikation im Frequenzraum wendest du die inverse Fourier-Transformation auf das Produkt an, um das weichgezeichnete Bild zurück im Ortsraum zu erhalten.

Die Nutzung des Faltungstheorems macht die Anwendung des Weichzeichnungsfilters oft schneller, besonders bei großen Bildern, weil Multiplikationen im Frequenzraum effizienter durchgeführt werden können als Faltungen im Ortsraum.

Was ist der Dirac Impulis?

Der Dirac-Impuls, symbolisiert durch $\delta(t)$, ist keine Funktion im herkömmlichen Sinn, sondern eine sogenannte Distribution oder verallgemeinerte Funktion. Hier sind einige Schlüsseleigenschaften:



(a) Dirac Impulse

1. Unendliche Amplitude: Der Dirac-Impuls hat zum Zeitpunkt $t = 0$ eine "unendliche" Amplitude, das heißt, er steigt an diesem Punkt auf einen unendlich hohen Wert.

2. Null Breite: Abgesehen vom Punkt $t = 0$ ist der Wert des Dirac-Impulses überall null. Die "Breite" des Impulses ist also Null.

3. Einheitsfläche: Trotz seiner unendlichen Amplitude und Null Breite hat der Dirac-Impuls eine Gesamtfläche (das Integral über den Impuls) von genau Eins. Mathematisch ausgedrückt:

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1$$

4. Auswahl-Eigenschaft: Eine der wichtigsten Eigenschaften des Dirac-Impulses ist seine Fähigkeit, Werte aus einer Funktion herauszuziehen. Wenn man den Dirac-Impuls mit einer anderen Funktion $f(t)$ faltet, erhält man die Funktion selbst zum Zeitpunkt Null.

In der Praxis wird der Dirac-Impuls verwendet, um ein ideales, momentanes Ereignis darzustellen, wie zum Beispiel einen Schlag auf eine Trommel oder einen elektrischen Schlag, der in einer unendlich kurzen Zeitspanne passiert.

Anwendung in der Computergraphik:

Textur-Mapping:
Beim Textur-Mapping kann der Dirac-Impuls dazu dienen, das ideale Sampling einer Textur zu beschreiben. Wenn eine Textur auf eine 3D-Fläche projiziert wird, müssen die Farbwerte der Textur auf die Pixel des Bildschirms abgebildet werden. Hierbei wird implizit eine Abtastung durchgeführt, die in ihrer idealen Form einem Dirac-Impuls entspricht, um den exakten Farbwert an einem bestimmten Punkt zu erfassen.

Rekonstruktion:
Der Dirac-Impuls wird auch in der Rekonstruktion von Signalen verwendet. Wenn eine Szene mit hoher Auflösung auf eine niedrigere Auflösung herunterskaliert wird, kann der Dirac-Impuls helfen, den Prozess des "Point Sampling" zu modellieren, bei dem nur der Farbwert des zentralen Punktes für das downgesampelte Pixel verwendet wird.

Erkläre Convolution (Faltung). Was hat es mit Box- und Tent aufsich?

Convolution (Faltung) kombiniert zwei Funktionen (oder Signale) miteinander, um eine dritte Funktion zu erzeugen. Dies geschieht durch Multiplizieren und Integrieren der Werte der einen Funktion mit einer umgekehrten und verschobenen Version der anderen Funktion über einen bestimmten Bereich. In der Computergrafik wird dies oft genutzt, um Filtereffekte wie Weichzeichnung, Schärfung oder andere Bildtransformationen zu implementieren.

1. Glättung des Signals:

Durch die Faltung eines Bildes oder Signals mit einem Glättungsfilter, wie dem Boxfilter, werden die scharfen Kanten und das Rauschen im Bild reduziert. Das Ergebnis ist ein weicheres Bild. Dieser Prozess kann als Low-Pass-Filterung betrachtet werden, da er hohe Frequenzen (Details und Rauschen) abschwächt.

2. Frequenzraum-Operationen:

Die Faltung im Ortsraum entspricht einer Multiplikation im Frequenzraum. Durch die Anwendung der Fourier-Transformation auf die beteiligten Funktionen und deren Multiplikation im Frequenzraum kann die Faltung effizient durchgeführt werden. Dies ist besonders nützlich für große Datenmengen, da die Faltung direkt im Ortsraum rechenintensiv sein kann.

3. Anwendung von Filtern:

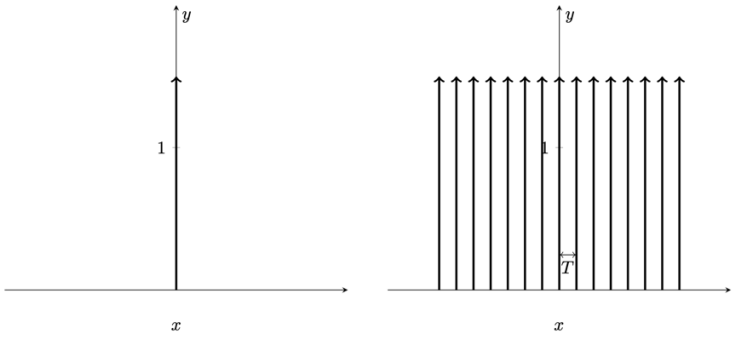
- ***Box-Filter:***
  Bei der Anwendung eines Boxfilters werden nur die Signale innerhalb der "Box" betrachtet, was bedeutet, dass benachbarte Werte interpoliert werden, um den Durchschnitt zu berechnen. Dies führt zu einer Glättung des Signals, indem hohe Frequenzen entfernt werden. Die Fouriertransformation eines Boxfilters ist die Sinc-Funktion, welche zeigt, wie Frequenzen im Signal durch den Filter beeinflusst werden.
- ***Tent-Filter:***
  Ein Tent-Filter verwendet lineare Interpolation zwischen Abtastwerten, was zu einer sanfteren Glättung führt als der Boxfilter. Die lineare Interpolation hilft, das ursprüngliche Signal besser zu approximieren, indem sie eine direktere Verbindung zwischen den Punkten herstellt.
- ***Sind das nicht dieselben Filter wie bei Reconstruction?***
  Ja und Nein. Die Filter selbst, also die mathematische Definition oder die Form der Filter (Boxform vs. Dreiecksform), sind dieselben. Der Unterschied liegt in ihrer Anwendung:

  Bei der *Convolution* (Faltung) geht es darum, ein bestehendes Signal oder Bild zu modifizieren, indem ein Filter darübergelegt wird, um bestimmte Effekte wie Glättung oder Schärfung zu erzielen.

  Bei der *Reconstruction* geht es um die Wiederherstellung oder das Nachbilden eines kontinuierlichen Signals aus einer Reihe von diskreten Abtastwerten, wobei die Filter dazu dienen, zwischen den Abtastpunkten zu interpolieren und das Signal zu glätten.

4. Spektrale Replikate:

Bei der periodischen Abtastung eines Signals (z.B. mit einer Kammfunktion) und dessen Transformation in den Frequenzraum können spektrale Replikate entstehen. Diese sind Kopien des ursprünglichen Signalspektrums, die in regelmäßigen Abständen im Frequenzraum auftreten. Sie entstehen aufgrund der Natur der Fourier-Transformation und der periodischen Wiederholung des Signals im Frequenzbereich. Spektrale Replikate müssen oft durch geeignete Filterung (z.B. mit einem Low-Pass-Filter) behandelt werden, um Aliasing-Effekte zu vermeiden und eine korrekte Rekonstruktion des Signals zu gewährleisten.



(a) Dirac Impulse (b) Comb Function

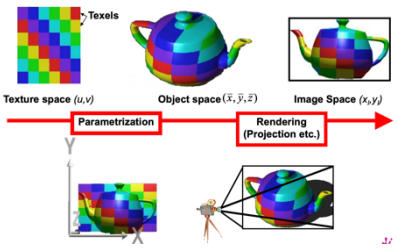
5. Frequency Domain:

Die "Frequency Domain" (Frequenzbereich) bezieht sich auf die Darstellung eines Signals oder einer Funktion in Bezug auf Frequenzen, statt in Bezug auf Zeit oder Raum. Dieser Wechsel der Perspektive wird typischerweise durch die Fourier-Transformation ermöglicht, die ein Signal aus dem Zeit- oder Ortsbereich ("Time Domain" oder "Spatial Domain") in den Frequenzbereich überführt.

Texturierung

Welche Arten von Texturierung gibt es? Was ist Parametrisierung?

Texturierung wird verwendet, um Oberflächen von 3D-Modellen detaillierter und realistischer erscheinen zu lassen. Dabei werden Texturen – Bilder oder Muster – auf die Oberflächen der 3D-Modelle projiziert. Es gibt verschiedene Arten der Texturierung, jede mit ihren eigenen Techniken und Anwendungen, um unterschiedliche visuelle Effekte zu erzielen.



Arten von Texturierung

- *Diffuse Texturierung:*
  Die grundlegendste Form der Texturierung, die einem Objekt Detail und Farbe verleiht. Die diffuse Textur repräsentiert die Grundfarbe des Materials.
- *Prozedurale Texturierung:*
  Anstatt vordefinierte Bilder oder Muster zu verwenden, generiert diese Technik Texturen durch Algorithmen. Prozedurale Texturen können endlos variieren und benötigen oft weniger Speicherplatz.
- *Abgetastete (Sampled) Texturen:*
  Diese Texturen werden durch das Abtasten realer Objekte oder Szenen mittels Digitalkamera, Scanner oder durch synthetische Erstellung gewonnen. Sie werden als Rasterbilder gespeichert und bieten eine hohe Vielfalt und Detailtreue. Die Herausforderung hierbei liegt in der korrekten Abbildung dieser 2D-Bilder auf 3D-Objekte, was durch Techniken wie UV-Mapping erreicht wird.

Parametrisierung

Parametrisierung bezieht sich auf den Prozess, durch den die 2D-Textur auf die 3D-Oberfläche eines Modells abgebildet wird. Dies erfordert die Definition einer Beziehung zwischen den 2D-Texturkoordinaten (üblicherweise u und v) und den 3D-Koordinaten des Modells. Das Ziel ist es, eine Methode zu schaffen, durch die jeder Punkt auf der 3D-Oberfläche eindeutig einem Punkt in der 2D-Textur zugeordnet wird.

Die Herausforderung bei der Parametrisierung besteht darin, Verzerrungen zu minimieren und sicherzustellen, dass die Textur gleichmäßig über das Objekt verteilt ist. Häufige Techniken der Parametrisierung umfassen:

- *UV-Mapping:*
  Manuelle oder automatische Zuweisung von 2D-Texturkoordinaten zu den 3D-Oberflächen.
- *Sphärisches Mapping und Zylindrisches Mapping:*
  Automatische Mapping-Methoden, die basierend auf der Form des 3D-Objekts eine einfache Parametrisierung ermöglichen.

Parametrisierung ist ein kritischer Schritt in der Texturierung, da sie bestimmt, wie gut die Textur auf das 3D-Modell passt und wie realistisch das endgültige Bild aussieht.

Erkläre was Aliasing ist und wie man es verhindern kann

Aliasing bezeichnet das Phänomen, bei dem hochfrequente Signale (feine Details oder schnelle Bewegungen in einem Bild) bei unzureichender Abtastung als verzerrte oder falsch dargestellte Informationen erscheinen. Dies äußert sich häufig in Form von Treppeneffekten an schrägen Linien

oder Kanten (sogenannte "Treppchenbildung"), Moiré-Mustern in fein gemusterten Bereichen oder flimmernden Effekten in bewegten Bildsequenzen.

Aliasing tritt auf, weil digitale Bilder eine endliche Auflösung haben und somit die Realität nur in diskreten Schritten abbilden können. Wenn die Auflösung nicht ausreicht, um feine Details korrekt zu erfassen, kann das digitale Bild die Originalszene nicht genau reproduzieren. Laut dem Nyquist-Shannon-Abtasttheorem muss ein Signal mit einer Frequenz, die mindestens doppelt so hoch ist wie die höchste Frequenz des zu erfassenden Signals, abgetastet werden, um eine korrekte Rekonstruktion ohne Informationsverlust zu gewährleisten.

Vermeidung von Aliasing
Um Aliasing zu vermeiden oder zu minimieren, werden verschiedene Techniken eingesetzt:

- *Antialiasing:*
  Dies ist eine Gruppe von Techniken, die darauf abzielen, die visuellen Effekte von Aliasing zu mildern. Antialiasing-Methoden können sowohl in der Hardware als auch in der Software implementiert werden und umfassen:
- *Erhöhung der Auflösung:*
  Eine höhere Bildauflösung kann das Aliasing reduzieren, da feinere Details besser erfasst werden können. Dies ist jedoch oft mit höheren Rechen- und Speicheranforderungen verbunden.
- *Präfilterung:*
  Bevor ein Bild gerendert oder digitalisiert wird, kann eine Präfilterung angewendet werden, um die Frequenz der Bildinhalte zu reduzieren und so die Anforderungen des Nyquist-Shannon-Abtasttheorems besser zu erfüllen. Dies kann bedeuten, dass feine Details absichtlich geglättet oder entfernt werden, um Aliasing zu verhindern.
- *Mipmaping:*
  Eine Technik, die speziell in der 3D-Computergrafik für Texturen verwendet wird. Hierbei werden mehrere Versionen einer Textur in verschiedenen Auflösungen gespeichert. Die passende Auflösung wird basierend auf der Entfernung des Betrachters ausgewählt, was Aliasing-Effekte bei Texturen reduziert.

Indem man diese Techniken anwendet, kann man die visuellen Störungen durch Aliasing erheblich reduzieren, was zu einer glatteren und angenehmeren Darstellung in digitalen Bildern und Grafiken führt.

Wie sind die UV-Koordinaten an den Meshs gemappt?

UV-Mapping ist der Prozess, durch den 2D-Texturbilder auf die Oberflächen von 3D-Modellen projiziert werden. Dieser Prozess erfordert die Umwandlung der 3D-Oberflächen der Modelle in 2D, um die Textur effektiv darauf anwenden zu können. Die Koordinaten im 2D-Texturraum werden typischerweise als U und V bezeichnet (anstelle von X und Y), um Verwechslungen mit den 3D-Raumkoordinaten zu vermeiden).

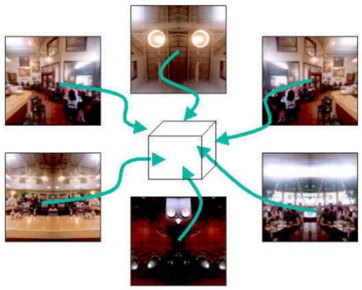
Funktionsweise des UV-Mappings

1. *Zuweisung von UV-Koordinaten:*

Jeder Vertex eines 3D-Modells wird mit einem Paar von UV-Koordinaten versehen. Diese Koordinaten definieren, wo auf der Textur der entsprechende Vertex abgebildet wird. Das UV-Mapping legt also eine direkte Beziehung zwischen einem Punkt auf dem 3D-Modell und einem Punkt auf der 2D-Textur fest.

2. *Erstellung der UV-Map:*
   Die UV-Map sieht aus wie eine ausgebreitete, flache Version der Oberfläche des 3D-Modells. Diese Map wird so angelegt, dass sie die Textur effektiv auf das Modell überträgt, während sie versucht, Verzerrungen zu minimieren.
3. *Texturierung:*
   Sobald die UV-Map erstellt ist, kann die Textur entsprechend den UV-Koordinaten auf das Modell angewendet werden. Jeder Punkt auf der Oberfläche des 3D-Modells holt sich seine Farbe (und möglicherweise andere Eigenschaften) von dem Punkt der Textur, auf den seine UV-Koordinaten verweisen.

Wie funktioniert Cube Mapping?



Cube Mapping ist eine Technik, die für die Erstellung von Umgebungsreflexionen, Refraktionen oder als Hintergrund für Szenen (bekannt als Skybox) verwendet wird. Diese Methode nutzt die sechs Seiten eines Würfels als die Projektionsfläche für eine Umgebungstextur, wobei jede Seite des Würfels ein Bild trägt, das zusammen eine 360-Grad-Ansicht der Umgebung bildet.

Funktionsweise des Cube Mapping

1. *Erstellung der Cube Map:*
   Zunächst wird eine Cube Map erstellt, indem sechs Texturen generiert oder zusammengestellt werden, die jeweils eine Ansicht der Umgebung aus der Perspektive des Mittelpunkts des Würfels in Richtung jeder der sechs Würfelseiten (oben, unten, links, rechts, vorne, hinten) darstellen.
2. *Texturierung:*
   Wenn ein Objekt in der Szene gerendert wird, berechnet der Shader für jeden Pixel des Objekts einen Reflexionsvektor oder einen Blickvektor, abhängig davon, ob Cube Mapping für Reflexionen oder Skybox-Effekte verwendet wird. Dieser Vektor gibt die Richtung an, in der der Pixel "schaut" oder reflektiert.
3. *Abfrage der Cube Map:*
   Der Reflexions- oder Blickvektor wird dann verwendet, um die entsprechende Position auf einer der sechs Seiten der Cube Map zu bestimmen. Die Texturkoordinaten innerhalb der Cube Map werden aus der Richtung dieses Vektors abgeleitet.
4. *Texturierung des Pixels:*
   Schließlich wird der Farbwert an dieser Position der Cube Map abgefragt und verwendet, um den Pixel auf dem Objekt zu texturieren. Dadurch entsteht der Eindruck, dass das Objekt seine Umgebung reflektiert oder dass eine Skybox die Szene umgibt.

Herausforderungen

- *Seamless Textures:*

Die Bilder auf den sechs Seiten der Cube Map müssen sorgfältig zusammengestellt werden, um sicherzustellen, dass die Übergänge zwischen den Seiten nahtlos sind, besonders bei der Skybox.

- *Limitationen bei dynamischen Szenen:*
  Da die Cube Map statisch ist, können dynamische Veränderungen in der Umgebung (wie bewegende Objekte oder wechselnde Lichtverhältnisse) nicht ohne weiteres dargestellt werden, es sei denn, die Cube Map wird regelmäßig aktualisiert.

Erkläre grob den Unterschied zwischen Bump, Parallax, Horizon und Relief Mapping.

Diese Techniken simulieren kleine Unebenheiten und Details, die das Licht in einer Weise brechen, die die Oberfläche realistischer erscheinen lässt.

Bump Mapping
Bump Mapping ist eine der ältesten und grundlegendsten Techniken, um Oberflächendetails zu simulieren. Es verwendet eine Graustufen-Textur (Bump Map), um zu bestimmen, wie stark das Licht an verschiedenen Punkten der Oberfläche abgelenkt werden soll, was den Eindruck von Erhebungen und Vertiefungen erzeugt. Bump Mapping ändert jedoch nicht die Silhouette des Objekts und ist am effektivsten bei Betrachtung aus der Ferne oder bei geringen Detailanforderungen.

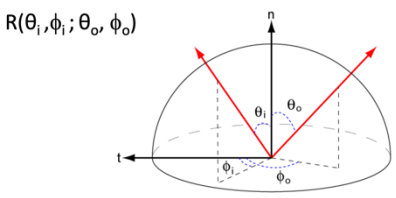
Parallax Mapping
Parallax Mapping ist eine Verbesserung des Bump Mapping, die eine stärkere Illusion von Tiefe erzeugt, indem sie die Sichtweise des Betrachters berücksichtigt. Es verschiebt die Texturkoordinaten basierend auf der Blickrichtung und der Position der virtuellen Kamera, um den Effekt von Tiefe zu simulieren. Parallax Mapping kann überzeugendere Ergebnisse als Bump Mapping liefern, besonders bei flachen Oberflächen, führt jedoch nicht zu einer Veränderung der Silhouette des Objekts.

Horizon Mapping
Horizon Mapping ist eine Technik zur Simulation von Selbstverschattung auf unebenen Oberflächen. Es basiert auf der Idee, die sichtbaren „Horizonte“, um jeden Punkt auf der Oberfläche herum zu berechnen, um zu bestimmen, wie Licht und Schatten über die Oberfläche fallen. Diese Technik kann zu einer realistischeren Darstellung von Unebenheiten führen, insbesondere bei komplexen Oberflächenstrukturen, ist aber rechenintensiver als einfache Bump- oder Parallax Mapping-Techniken.

Relief Mapping
Relief Mapping ist eine noch fortschrittlichere Technik, die eine deutlich überzeugendere Simulation von Tiefe ermöglicht, indem sie ein detailliertes Profil der Oberfläche in der Textur speichert und dieses Profil bei der Rendering-Zeit durch Ray-Casting oder ähnliche Verfahren auswertet. Diese Technik kann echte Geometrieveränderungen simulieren, einschließlich korrekter Selbstverschattung und Silhouettenänderungen. Relief Mapping ist jedoch deutlich rechenintensiver als die anderen genannten Techniken und wird in der Regel für hochwertige Visualisierungen verwendet, wo Detailtreue und visuelle Qualität Vorrang haben.

Lighting & Physically-Based-Shading:

Was ist BRDF? (grobe Erklärung, 4D Formel, Verhältnis Incoming/Outgoing)



Die Bidirektionale Reflexionsverteilungsfunktion (BRDF: Bidirectional Reflectance Distribution Function) ist ein Konzept, das beschreibt, wie Licht von einer Oberfläche reflektiert wird. Grob gesagt gibt die BRDF an, wie das Licht, das aus einer bestimmten Richtung auf eine Oberfläche trifft (incoming), in eine andere Richtung (outgoing) reflektiert wird. Die Funktion hilft zu verstehen und zu berechnen, wie hell eine Oberfläche erscheinen wird, wenn sie aus verschiedenen Winkeln beleuchtet und betrachtet wird.

4D Formel und Verhältnis Incoming-Outgoing

Die BRDF $R(\theta_i, \phi_i; \theta_o, \phi_o)$ ist eine Funktion, die die Reflexionseigenschaften einer Oberfläche definiert. Sie gibt an, wie viel Licht, das aus einer bestimmten Eingangsrichtung (gekennzeichnet durch die Winkel θ_i und ϕ_i) auf die Oberfläche trifft, in eine bestimmte Ausgangsrichtung (gekennzeichnet durch die Winkel θ_o und ϕ_o) reflektiert wird.

Hier ist eine grundlegende Erklärung der Parameter:

- θ_i: Der Einfallswinkel des Lichts – der Winkel zwischen der einfallenden Lichtstrahlrichtung und der Normalen der Oberfläche (n).
- ϕ_i: Der Azimutwinkel der einfallenden Lichtstrahlrichtung in der Ebene parallel zur Oberfläche.
- θ_o: Der Ausfallswinkel des reflektierten Lichts – der Winkel zwischen der reflektierten Lichtstrahlrichtung und der Normalen der Oberfläche (n).
- ϕ_o: Der Azimutwinkel der reflektierten Lichtstrahlrichtung in der Ebene parallel zur Oberfläche.

Physikalische Grundlagen und Anwendungen

Ein Schlüsselaspekt der BRDF ist die Energieerhaltung, was bedeutet, dass die Menge des von einer Oberfläche reflektierten Lichts nicht größer sein kann als die Menge des einfallenden Lichts. Dies ist wichtig, um realistische Renderings zu erzeugen, die physikalisch plausibel sind.

Die BRDF spielt eine zentrale Rolle beim Physically-Based Rendering (PBR), da sie hilft, Materialien basierend auf ihren physikalischen Eigenschaften realistisch darzustellen. Verschiedene Materialien haben unterschiedliche BRDFs – beispielsweise reflektiert ein glänzendes Material das Licht anders als ein mattes Material.

In der Praxis werden oft vereinfachte Modelle oder Annäherungen der BRDF verwendet, um die Berechnungen handhabbar zu machen, während gleichzeitig ein hohes Maß an Realismus beibehalten wird. Solche Modelle können Aspekte wie diffuse Reflexion (gleichmäßige Lichtstreuung), spekulare Reflexion (spiegelähnliche Reflexion) und mehr abdecken.

Erkläre den Unterschied zwischen Diffuse Reflektion / Spekulare Reflektion?

In der Computergrafik beschreiben diffuse Reflexion und spekulare Reflexion zwei verschiedene Arten, wie Licht von Oberflächen reflektiert wird. Diese Reflexionsarten sind entscheidend für das Verständnis und die Simulation der Weise, wie Materialien Licht absorbieren und reflektieren, und beeinflussen direkt, wie realistisch ein gerendertes Bild aussieht.

Diffuse Reflexion
Diffuse Reflexion tritt auf, wenn Licht auf eine raue oder matte Oberfläche trifft und in viele verschiedene Richtungen gestreut wird. Diese Art der Reflexion sorgt dafür, dass die Oberfläche aus allen Blickwinkeln gleichmäßig beleuchtet erscheint.

- *Eigenschaften:*
  Die Farbe einer diffus reflektierenden Oberfläche ist ein direktes Ergebnis des diffus reflektierten Lichts. Dies bedeutet, dass die Farbe des Objekts von jedem Winkel aus gleich aussieht, unabhängig davon, woher das Licht kommt oder wo der Betrachter steht.
- *Anwendung:*
  Diffuse Reflexion wird verwendet, um das Aussehen von Materialien wie Papier, Stoff, einigen Kunststoffen oder rauen Metallen zu simulieren.

Spekulare Reflexion
Spekulare Reflexion tritt auf, wenn Licht auf eine glatte oder glänzende Oberfläche trifft und in einer bestimmten Richtung reflektiert wird, ähnlich wie ein Spiegel. Die Richtung der Reflexion hängt vom Einfallswinkel des Lichts ab (nach dem Gesetz "Einfallswinkel gleich Ausfallswinkel").

- *Eigenschaften:*
  Spekulare Reflexion erzeugt helle Flecken oder Glanzlichter auf der Oberfläche, die sich bewegen oder ändern, wenn der Betrachter oder die Lichtquelle ihre Position ändern. Die Farbe des reflektierten Lichts kann von der Farbe der Lichtquelle oder der Farbe der Oberfläche selbst beeinflusst werden, abhängig von den Materialien.
- *Anwendung:*
  Spekulare Reflexion wird verwendet, um das Aussehen von Materialien wie Metall, Wasser, Glas und polierten Steinen zu simulieren.

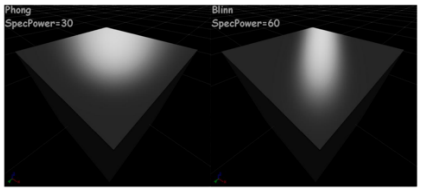
Was ist der Unterschied zwischen Phong & Blinn-Phong Modell?
Der Unterschied zwischen Phong-Shading und Blinn-Phong liegt in der Art und Weise, wie sie spekulare Reflexionen – die hellen Glanzlichter, die auf glänzenden Oberflächen erscheinen – berechnen. Beide Modelle werden verwendet, um die Wechselwirkung von Licht mit Oberflächen zu simulieren, aber sie nehmen unterschiedliche Annahmen und Berechnungen für den spekularen Anteil vor.

Phong-Reflexionsmodell

- *Berechnungsmethode:*
  Phong verwendet den Vektor der perfekten Reflexion, der die Richtung beschreibt, in die ein Lichtstrahl von einer Oberfläche perfekt reflektiert würde. Die Intensität der spekularen Reflexion wird dann durch die Berechnung des Winkels zwischen diesem perfekten Reflexionsvektor und dem Blickvektor zum Betrachter bestimmt. Die Berechnung nutzt die Kosinus-Potenz dieses Winkels, um die Glanzlichter zu modellieren.
- *Charakteristik:*
  Das Phong-Modell ist bekannt für seine Fähigkeit, sehr glatte und glänzende Highlights zu erzeugen, die gut zu vielen Arten von Materialien passen. Es kann jedoch rechenintensiv sein, insbesondere bei der Berechnung der perfekten Reflexionsvektoren.

Blinn-Phong-Modell

- *Berechnungsmethode:*
  Blinn-Phong modifiziert die Art und Weise, wie die spekulare Reflexion berechnet wird, durch Einführung des sogenannten "Halfway"-Vektors, der zwischen dem Lichtvektor (von der Lichtquelle zur Oberfläche) und dem Blickvektor (vom Betrachter zur Oberfläche) liegt. Die Intensität der spekularen Reflexion wird dann durch die Berechnung des Winkels zwischen diesem Halfway-Vektor und der Normalen der Oberfläche bestimmt.
- *Charakteristik:*
  Blinn-Phong ist effizienter in der Berechnung als das ursprüngliche Phong-Modell und liefert in vielen Fällen ähnliche Ergebnisse. Die Verwendung des Halfway-Vektors vereinfacht die Berechnungen und kann zu einer etwas weicheren Verteilung der Glanzlichter führen, was bei bestimmten Materialien realistischer wirken kann.



TLDR

- *Berechnungsgrundlage:*
  Phong nutzt den perfekten Reflexionsvektor, während Blinn-Phong den Halfway-Vektor zwischen Licht- und Blickrichtung verwendet.
- *Leistung:*
  Blinn-Phong ist im Allgemeinen effizienter in der Berechnung und wird oft als Optimierung des Phong-Modells angesehen.
- *Visuals:*
  Blinn-Phong kann zu einer weicheren Verteilung der Highlights führen, während Phong für seine Fähigkeit bekannt ist, sehr präzise und glänzende Glanzlichter zu erzeugen.

Was ist Ambient Illumination?

Ambient Illumination (Umgebungsbeleuchtung) beschreibt eine einfache, gleichmäßige Beleuchtung, die in einer Szene auf alle Objekte wirkt, unabhängig von ihrer Position oder Ausrichtung. Diese Art der Beleuchtung simuliert das diffuse, indirekte Licht, das von Oberflächen innerhalb einer Szene reflektiert wird und überall vorhanden ist, nicht von einer spezifischen Lichtquelle stammt.

Die Idee hinter Ambient Illumination ist, die Reflexion aller indirekten Beleuchtungen zu repräsentieren – also das Licht, das von den Oberflächen im Raum mehrfach gestreut wird, bevor es ein Objekt erreicht. In der realen Welt ist solches Licht für die Sichtbarkeit in Schattenbereichen und für die weichen Übergänge zwischen beleuchteten und nicht direkt beleuchteten Bereichen verantwortlich.

Da die genaue Berechnung aller möglichen Lichtwege, die Global Illumination (globale Beleuchtung) erfordert, rechenintensiv ist, dient Ambient Illumination als eine Art "Hack" oder Näherung, um ohne diesen Aufwand ein grundlegendes Beleuchtungsniveau zu gewährleisten.

Limitationen:
Ambient Illumination allein kann keine realistischen Beleuchtungseffekte wie weiche Schatten, die Interaktion von Licht mit verschiedenen Materialien oder die subtile Variation von Licht in einer Szene erzeugen.